A Novel Approach for Data Encryption in Hadoop

Girish Prasad Patro

Roll. 710cs2119

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769008, India

A Novel Approach for Data Encryption in Hadoop

Dissertation submitted in

MAY 2015

to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Master of Technology

Under the Dual Degree Programme

by

Girish Prasad Patro

(Roll. 710CS2119)

under the supervision of

Prof. Sanjay Kumar Jena

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela – 769 008, India

Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela-769 008, India. www.nitrkl.ac.in

Declaration

I certify that,

- I have complied with all the benchmark and criteria set by NIT Rourkela Ethical code of conduct.
- The work done in this project is carried out by me under the supervision of my mentor.
- This project has not been submitted to any other institute other than NIT Rourkela.
- I have given due credit and references for any figure, data, table which was being used to carry out this project.

Date: 01 June 2015 **Girish Prasad Patro**

Place: Rourkela

Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela-769 008, India. www.nitrkl.ac.in

Certificate

This is to certify that the work in the thesis entitled ***A Novel Approach for Data Encryption in Hadoop*** by ***Girish Prasad Patro***, having roll number ***710CS2119***, is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of ***Master of Technology, Under the Dual Degree Programme*** in ***Computer Science and Engineering Department***. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Date: 01 June 2015 **Dr. Sanjay Kumar Jena**

Place: Rourkela

Acknowledgment

All the efforts I have put forward in carrying out this project would have been in- complete, if not for the kind support of many individuals as well as this institute. I would like to express my deep sense of gratitude to all of them. Foremost, I would like thank my supervisor of this project, Prof. Sanjay Kumar Jena, Department of Computer Science and Engineering, National Institute of Technology, Rourkela, for his incalculable contribution in the project. He stimulated me to work on the topic and provided valuable information which helped in completing the project through various stages. I would also like to acknowledge his exemplary guidance, monitoring and constant encouragement throughout the course of this thesis.

I am obliged to all the professors of the Department of Computer Science and Engineering, NIT Rourkela for instilling in me the basic knowledge about the field that greatly benefited me while carrying out the project and achieving the goal. I thank the admin and staff of National Institute of Technology Rourkela for extending their support whenever needed.

I also thank my friends and peers who have extended their help whenever needed. Their contribution has always been significant. Finally, I would like to take this opportunity to thank my parents, who have always been a source of inspiration and motivation for me, and also for the love they have provided in stressful periods, which has been a guiding force for the completion of the thesis.

Girish Prasad Patro

Abstract

The data used by various organizations and institutions is increasing at a rapid rate. In todays world, organizations have to process petabytes of data. The traditional database management system fails to process such large amount of data. So, we need to find out an effective way of approach for handling and processing such large amount of data which gives rise to big data problem. Size or volume of the data is not only the single criteria to classify big data, we have to keep in mind the type of the data that is whether data is structured or semistructured or unstructured. Semistructured or unstructured data is difficult to manage through traditional database management system. To overcome this big data problem Apache produced a software tool named Hadoop which uses MapReduce architecture to process large amount of data. MapReduce architecture processes the data in parallel manner. Hence, we have analyzed the performance of Hadoop with increase in size of the dataset. We have analyzed how Hadoop's performance varies with increase in size of the dataset. Though it gives solution to overcome the big data problem but it does not ensure the security and privacy of the stored files in Hadoop. In this dissertation, an asymmetric key cryptosystem is proposed for the encryption of files stored in HDFS. We used our proposed cryptosystem to encrypt the content of data before storing it in HDFS. The results obtained from various experiments indicate favorable results of above approach to address security problem associated with big data.

***Keywords*:** Big Data, Hadoop, MapReduce, Encryption, Security

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid increase in size of data has now emerged as the biggest problem for many large organization and institutions. For Example, let us take an example of Google or Facebook which deals with petabytes of data daily. In Facebook, starting from creating profile to like, comment, sending friend request, adding visited places, messages every details has to be stored in database so that whenever needed by the user, it can be available. Same is the case with Google which has to store all the information regarding the user mail account, the search items done by the user, the last visited sites by the user etc. Similarly all companies deals with vast amount of data which given rise to big data problem.

Big data has emerged as a challenge for the researchers. Their main focus is how to handle such vast amount of data, how to process those large amount of data with in a acceptable time limit. The traditional database management system also can be used but this is not a feasible option to tackle the big data problem. Generally the traditional database management system has a size cap as how much data it can process. If this is the case, then it is absolutely impossible to overcome the problems faced by the big data which grows exponentially. Moreover, some database supports infinite data but the data accessing time and processing time is extremely large which is not feasible. So we have to find an efficient solution for these problems or challenges faced by big data problem.

The main focus will be to execute this large size of data efficiently with in a desirable time limit. Many researcher have proposed many methods to overcome such problem. Doug Cutting and Mike Cafarella invented Hadoop and the name became Hadoop after his son 's toy elephant [1]. The Nutch search engine project was first supported by this Hadoop. Later on it was acquired by apache and was made open source.

In this dissertation, we have first analyzed the performance of Hadoop based on criteria such as with increase in size of the data set. We increased the size of the data set in each experiment to find out how Hadoop performs under varying size. As Hadoop does not come with any security, so the data is not safe and its security is at stake. Hence, we have implemented our cryptosystem to files before storing it in HDFS and then compared this with previous or normally used Hadoop setup.

1.1 Motivation

With the advent of technology the data is also increasing at a rapid rate in this digital world where everything goes electronically. So now the focus is on how to address this big data problem. Continuous research has been done regarding the big data problem and how to develop a parallel processing architecture to solve it. This continuous research resulted in invention of Hadoop which processes the data in sequentially manner not like the traditional system which processes the data in parallel manner. Hence, Hadoop can be useful as a tool to overcome the size of the dataset. Moreover, though Hadoop solves issues regarding big data like processing big data, managing big data with in a tolerable time limit etc, it still lacks in security aspect. The security and privacy of data is not ensured. This did motivate us to apply security mechanism in Hadoop and its performance evaluation.

1.2 Problem Statement

The main focus will be the performance analysis of Hadoop to find out how it handles big data, whether it is feasible or not, How Hadoop reacts with increase in size of the data set. Furthermore, Hadoop does not come with any security features. So we have proposed an asymmetric key cryptosystem to apply encryption to files before storing it in Hadoop. Then we compared our work the existing one.

1.3 Related Work

Hadoop is a software paradigm which is designed on MapReduce architecture. MapReduce executes the job parallely in two phases that is map and reduce. In paper [2], the author discussed how traditional database is not a scalable approach for big data problem and it needs to be improved. HDFS is an open source undertaking of Google appropriated record system(GFS). In paper [3], the author discussed what are the difficulties, problems and challenges associated with big data. It has a high adaptation to internal failure and certain information access control. We principally utilize Hadoop's HDFS (Hadoop Distributed File System) for distributed storage. As Hadoop likewise needs security measures, Kerberos was coordinated into the Hadoop in 2009 by hurray. The client need to acquire access affirmation from the outsider for key issues before accessing to Hadoop [4]. But this kerberos has its own drawbacks and the security can be easily compromised by an attacker. This motivated us to discover an efficient cryptosystem and apply the same to files before storing it in HDFS and its performance evaluation.

1.4 Contributions

1. Performance analysis of Hadoop has been carried out with increase in size of dataset.

2. A novel encryption scheme has been proposed which is an asymmetric key cryptosystem to encrypt files before storing it in Hadoop.

1.5 Thesis Organization

In this chapter, we have discussed the motivation behind carrying out the performance evaluation of Hadoop as well as security concerns for the files stored in Hadoop. The organization of the rest of the thesis and a brief outline of the chapters in this thesis are as given below.

In Chapter 2, we have given a description about big data which includes what is big data, what is big data problem, how to face the challenges caused by big data, what is Hadoop, different components of Hadoop etc.

In Chapter 3, we have discussed performance evaluation of Hadoop with increase in size of the data set and the corresponding result is shown.

In Chapter 4, we have discussed security concerns for Hadoop. This includes vulnerabilities, threats for Hadoop, existing methods to overcome this problem.

In Chapter 5, we have implemented our proposed model and the result is taken. Then we compared the obtained result with existing technology.

In Chapter 6, we concluded with the result which favored our method over existing method for applying security in Hadoop.

Chapter 2

Big Data Overview

2.1 Big Data

Big data is an expensive term used for very large data sets which is very complex in nature. It is so big in size that traditional or conventional data processing applications are inadequate. The big data comes with various challenges which includes analysis, search, finding pattern, capture, sharing, storage, security, theft issues, transfer, information privacy and visualization [5]. The big data often refers normally to many advanced methods to extract pattern or value from data and the use of predictive analytics . At sometimes, it refers to a particular size of data set and sometimes the nature of the dataset. It is very important to have accuracy in big data which helps us in confident decision making. Better the decisions, greater the cost reductions, operational efficiency and reduced risk.

The exponential growth of information or data throughout the most recent decade has presented another space in the field of data innovation called big data. Datasets that extends the breaking points of customary information handling and stockpiling frameworks is frequently alluded to as big data. The need to process and dissect such enormous datasets has given rise to big data Analysis. Big data includes Enormous information or data generated by large organization or companies which consists of

unstructured and semistructured data. The information that would take a lot of time and cost an excessive amount of cash to load into a database for examination, falls under big data category.

2.2 Big Data Problem

Big Data has developed on the grounds that we are using many kind of technologies which uses very large amount of data. One present challenge associated with big data is the trouble we face while working with it by utilizing traditional statistics/visualization packages and relational databases. So, we need a "greatly parallel programming running on many number of servers". The other different difficulties confronted in big data administration includes adaptability, unstructured data, availability, accessibility, constant monitoring, fault tolerance and many more. Notwithstanding varieties in the measure of data put away in different sectors, the sorts of data produced and put away i.e., whether the data encodes pictures, audio, video, or content/numeric data additionally vary notably from industry to industry [6]. So, we need a mechanism to overcome such big data problem.

2.3 Hadoop

Hadoop is a software paradigm written in Java, circulated under Apache License. The attributes of Big Data can be comprehensively partitioned into four "Vs"i.e. Volume, Velocity, Varity and Variability [7]. Volume alludes to the extent of the information. While Velocity tells about the pace at which information is produced; Varity and Variability lets us know about the multifaceted nature and structure of information and distinctive methods for translating it. A brief detail of these four vs are important.

1. **Volume**: Hadoop gives system to scale out evenly to self-assertively substantial information sets to address such huge volume of information.

2. **Velocity:** Hadoop handles incensed rate of approaching information from expansive framework.

3. **Variety :** Hadoop bolsters complex occupations to handle any mixture of unstructured information.

4. **Variability :** Hadoop handles data which is complex and varies according to size.

Figure 2.1: 4 Vs of Big Data

Figure 2.1 shows the 4 "V"of big data as discussed earlier. These 4 Vs defines big data.

The key attributes of Hadoop are that it is redundant and reliable, that is if you lose a machine due to some failure, it automatically replicates your data immediately without the operator having to do anything, it is extremely powerful in terms of data access and is preliminary batch processing centric and makes it easier to distributed applications using MapReduce software paradigm. Moreover it runs on commodity

hardware which cuts off the cost of buying special expensive hardware and RAID systems [8].

In conventional methodology, we used to have a capable PC to process the available data. Anyway, there is a furthest point of confinement to the measure of data prepared in light of the fact that it is not adaptable and big data develops with incredible speed and variety. But Hadoop takes after an altogether different approach when contrasted with the customary undertaking methodology. First, big data is broken into small chunks so that vast measure of data can be taken care of efficiently and successfully. Alongside this information isolation, Hadoop does the calculations or execution of program in parallel. When all the kinds of calculations are done at the slave level, the outcomes are merged or grouped together and sent back to the application.

2.3.1 HDFS

HDFS stands for Hadoop Distributed File System which is a distributed filesystem. It is designed so that very large size files can be stored with streaming data access patterns, running on clusters of commodity hardware.

1. **Streaming data access**

   HDFS is designed with keeping in mind that we can use it like reading many times but writing only once. After accepting a dataset, we have to analyze it thoroughly to understand the hidden pattern. So we are dealing here how to read the entire data set not just part of it.

2. **Commodity Hardware**

   Hadoop can be made run on low cost, easily available systems. We usually run it on nodes of commodity hardware. As, cluster failure can happen, Hadoop is designed in such a way that it can overcome such failure easily [9].

2.3.2 Fault tolerance

Hadoop is constructed keeping in mind that hardware failure or software failure may occur. As Hadoop is configured across nodes so it is necessary to keep track of each node continuously. If one or more node fails, it is detected by Hadoop as the job assigned to Jobtracker is to regular monitoring the DataNodes and send report to NameNode [8]. So if such failure occurs, the task assigned to that corresponding node will be reassigned to some other nodes. Similarly the job of secondary NameNode is to keep track of the NameNode. If failure of NameNode occurs, the secondary NameNode plays the role of NameNode and start keeping information about the DataNodes. In this way fault tolerance is done by Hadoop.

2.3.3 Replica Placement

The placement of replica is critical to HDFS reliability and performance. HDFS goes about as a self-recuperating framework. If one of the DataNode fizzles, we still have two different DataNodes having the needed information's reproductions. On the off chance that a DataNode goes down, then the pulse from DataNode to NameNode will stop and following ten minutes NameNode will consider that DataNode to be dead and all the obstructs that were put away on that DataNode will be rereplicated and distributed equally on other active DataNodes.

2.3.4 Rack Awareness

A HDFS record comprises of blocks. At whatever point another block is obliged to store the data, the NameNode designates a place with a block ID and decides a list of DataNodes to host replicas of the data. Data is then pipelined from the customer to the DataNodes in taking the minimum distance from the customer. Nodes are spread over numerous racks that share a switch associated by one or more center switches. The NameNode, going about as a focal spot keeps up the metadata that aides in determining the rack area of every DataNode [8]. The primary intention

of rack awareness policy is to enhance reliability and accessibility of data alongside network bandwidth utilisation. The main objective of Rack Awareness strategy is as per the following:

1. No DataNode ought to contain more than one copy of any block of record.

2. No rack ought to contain more than two copies of the same block, given there are adequate quantities of racks on the cluster.

2.4 MapReduce

Hadoop Mapreduce is a software programming model to process large datasets on clusters in an efficient manner. A MapReduce task divides the input data into number of chunks which are assigned to different DataNodes and are processed in parallel manner. The input data is given as input into map function which results in another set of <key,value>pairs. This another set is normally considered as intermediate pair. The output of the above map function is sorted automatically. The output of the map function consists different values associated with the corresponding key. And after the result of the map function is sorted, it is fed into reduce function. The job of reduce function is to reduce all ¡key,value¿ pairs which is obtained from the map function [10]. The MapReduce framework splits the data across several DataNodes which schedules the execution of task along with cluster-cluster communications. If any DataNode failure occurs it can be detected by NameNode which continuously monitors the DataNodes. There comes the role of replica where the failure node data can be given to other active DataNodes and the execution continues in parallel manner.

- **"Map"step :**In Map step the master node gets the input whose ¡key, value¿ is normally ¡line number, record¿ but we can decide the key-value pair as per our requirements. After getting the input, map function does partition of data and convert into no of small chunks which are usually of size 64MB. After the

chunk creation Hadoop distributes the chunks to slave nodes. A slave node may have to do the partitioning and distributing as per the client's requirement which leads to a multi-level hierarchical structure. Now comes the role of slave nodes. The objective of slave node is to processes the no. of chunks received from master node and to send the result back to its master node. Map function actually receives one pair <key, value>of data and produces another pair <key, value>of data which is normally called intermediate <key,value>pair.

$$\textbf{Map (k1, v1)} \implies \textbf{list (K2, v2)}$$

- **"Reduce"step :**After the slave node sends the result back to master node, the job of master node is to collect those result received from slave nodes and to combine those results to form the desired output. So basically reduce function takes the intermediate <key, value>pair obtained from map function and combines them or groups them and performs instructions as per the requirement. The job of Reduce function is to apply the desired operations on each chunk of the data in a parallel manner. As a result it produces the desired output.

$$\textbf{Reduce (K2, list (v2))} \implies \textbf{list (v3)}$$

2.5 Hadoop Components

Hadoop consists of various components such as NameNode, DataNode, Secondary NameNode, Node Manager, Resource Manager. We will discuss each component of Hadoop in more details.

2.5.1 Namenodes and Datanodes:

A HDFS group has two sorts of nodes working in an expert specialist design:

- A NameNode

- Various DataNodes

When the job is assigned, the Namenode checks how many Datanodes are active and free to take up the job. The Namenode then assigns the job to such Datanodes and keep track of all the Datanodes that is it continuously communicate with all the datanodes to check its status, job's status so that if any failure occurs, it can assign the job to other active and free Datanodes.

Datanodes actually perform the jobs assigned to them by the Namenode. It continuously sends the status to Namenode to let the Namenode that it is active and the task is still on. Once the job is complete, it sends the status as complete and waits to take up another job.

The filesystem is worth of nothing without the use of NameNode. The filesystem is meaningless without NameNode. Surely, if the server in which the NameNodes are running were pulverized, we would lost all the records which are available on the filesystem. The data would be lost because it is highly unlikely to get the data back that is we cant rebuild the data or records which has been lost in this process. Thus, it is basic to make the NameNode sufficiently solid to overcome disappointment. Hadoop gives two approaches to prevent such kind of issues [9].

2.5.2 ResourceManager

The ResourceManager has two main components:

- Scheduler
- ApplicationsManager

The objective of Scheduler is to allot resources to the every single running application remembering the well known requirements of queues, capacities etc. The Scheduler is capable as in it performs no checking of use or following of

status for the application. Additionally, it doesn't offer any certifications about restarting the tasks which has failed either because of hardware failure or software failure. In the first form, just memory is supported. The Scheduler has a pluggable arrangement module, which is in charge of distributing the cluster resources among the different queues, applications etc. The present MapReduce schedulers, for example, the CapacityScheduler and the FairScheduler would be a few illustrations of the module. The CapacityScheduler backings hierarchical queues to consider more unsurprising sharing of cluster resources. The ApplicationsManager is in charge of tolerating job entries, arranging the first container for executing the application particular ApplicationMaster and gives the support of restarting the ApplicationMaster container on failure [11].

2.5.3 NodeManager

The NodeManager is the per-server system specialists who is in charge for checking their resource utilization (disk, cpu, network) and reporting the same to the ResourceManager/Scheduler. It is also responsible for container. The ApplicationMaster has the obligation of arranging proper resource container from the Scheduler, following their status and checking for advancement.

2.5.4 Secondary Namenode

Secondary NameNode comes into action when the primary NameNode fails in any case. Secondary NameNode keep tracks of the NameNode and communication between NameNode and DataNode occurs spontaneously. If the NameNode fails, the communication between the NameNode and secondary NameNode is broken. Then the secondary NameNode plays the role of NameNode.

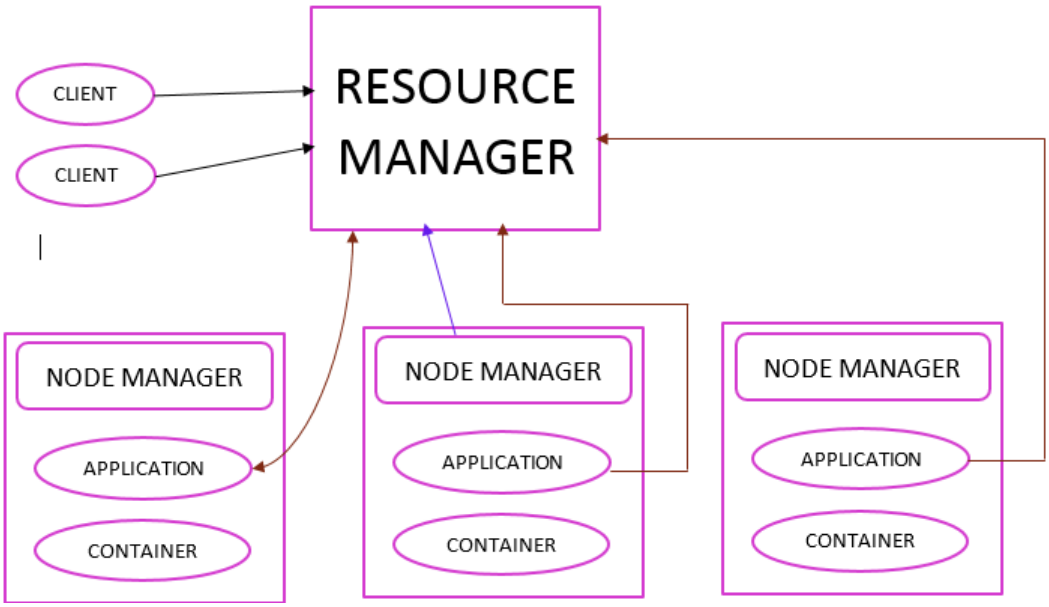

Figure 2.2: Hadoop Component

Figure 2.2 shows different components of Hadoop. We can see that how each components are related to each other. Clients first requests for resources to the resource manager. According to availability, resource is allocated. NodeManager constantly monitors the status of its corresponding node and sends the result to ResourceManager. If any failure occurs at any node, then the communication line between that node and the ResourceManager cutoff. So the ResourceManager allocates the job of that node to other active nodes.

Chapter 3

Performance Evaluation

3.1 Introduction

We have learned about every components of Hadoop. Now we will find out if using Hadoop is a scalable approach or not while addressing big data problem. As big data problem comprises of processing a very huge amount of data within a tolerable time which cant be achieved by using traditional database management system. As Hadoop provides a distributed architecture to carry out the assigned task, we have analyzed its performance with increase in size of the dataset. Moreover, we have carried out another performance analysis test with increase in number of nodes.

3.2 Experimental setup

The experiment is carried out on a system having configuration of intel core i5-480M Processor 2.66GHz with Turbo Boost up to 2.93GHz with 4GB RAM. Hadoop-2.4.0.tar is installed on ubuntu-12.04.3. Single node architecture of Hadoop is used in which all components of the Hadoop are on the same node. Hadoop requires JDK for its working. Java OpenJdk is installed on this system using the apt-get command. The running component of the Hadoop can be checked using the jps command. The output of the command is shown in Figure4. We can access that

interface from http//:localhost:50070.

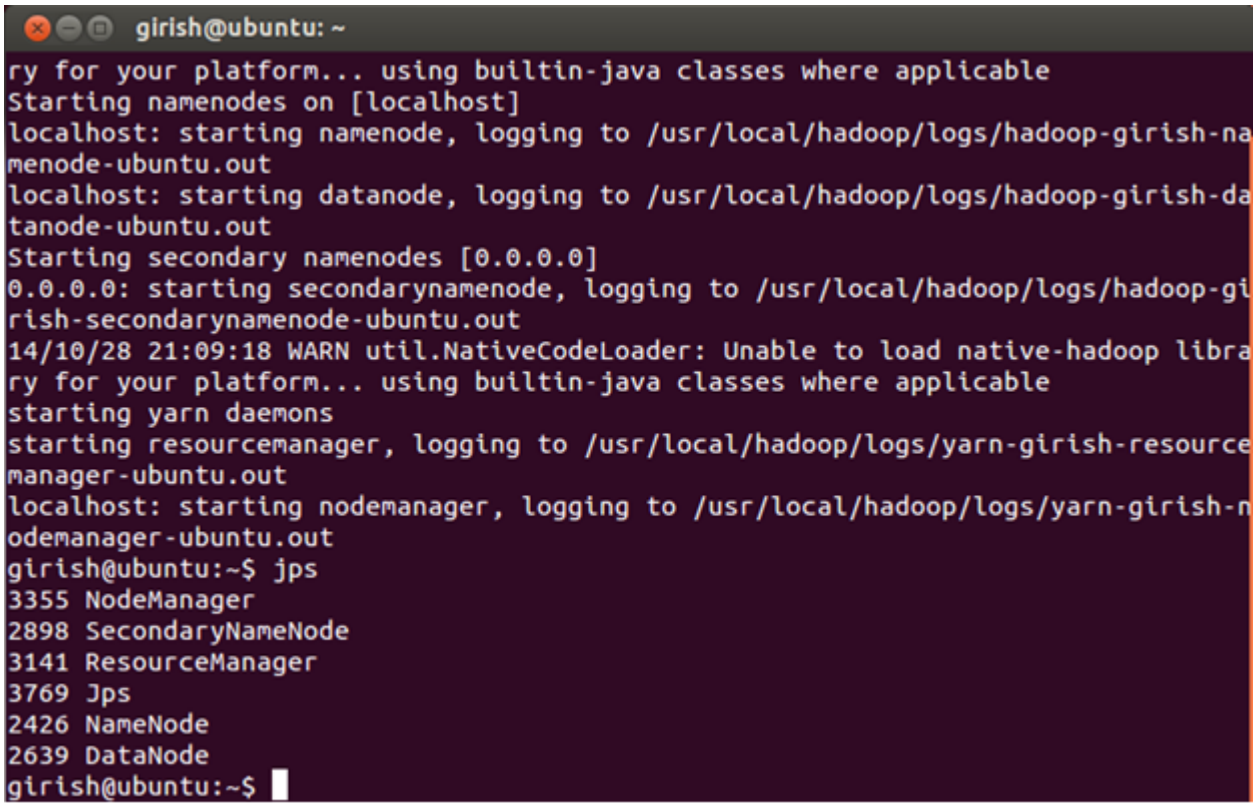

Figure 3.1: output of jps command

Figure 3.1 shows the status of different components of Hadoop.

3.3 Data Set

We have taken the million song dataset which is freely available on internet which is provided by the labrosa.ee.columbia.edu . Below is a list of all the fields associated with each track in the database which shows the various fields of the dataset and which field represents what.

To help in research work they also provide a subset of dataset which consists of less songs than the original dataset and also lesser in size. So we have taken the dataset in different size and carried out the experiment. We have increased the size

in our experiment as we proceed. We have analyzed different attributes according to the result of the experiment.

The various fields are separated by commas as the dataset which is downloaded is in cvs format. After running the program in map step the different field values play the roles of keys and each key is associated with a value of 1. So the output of the map function consists of <field value,1>for each record. Such as track_id, song_id,1, artist name, song title etc. As our objective is to find out the number of tracks done by each artist and to list out those tracks, So we will be dealing with song_title and artist_name.

3.4 Result of MapReduce function

The map function will take each record as input. The key will be the line number and value will be each record. So the <key,value>pair for the input of our map function will be <line_number,Record>. The map function will produce the output as considering key as Artist_name and value as song_title. So the output of the map function will be <Artist_name,song_title>. The MapReduce architecture will process the data in parallel manner in its clusters and produce output with key as artist_name and value as Song_titles array. In our case, it will group different song_titles according to artist_name.

After getting the input, the reduce function will merge the data according to key value. For each artist_name the counter will count the number of song_titles written by the artist and will list out those song_titles. So for example, if a artist_name has 15 song_titles, it will first show that the artist has written 15 song_titles and after that it will list out all the song_titles.

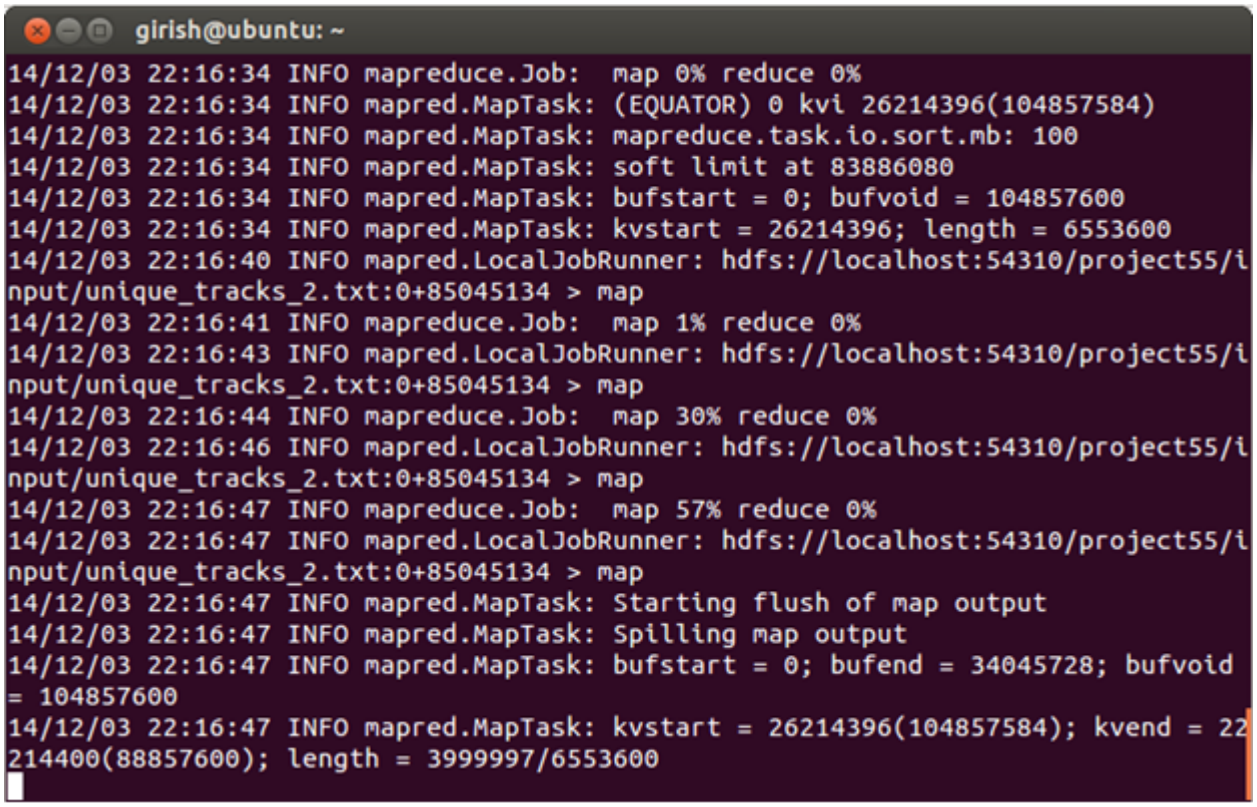

Figure 3.2: MapReduce

Figure 3.2 shows how the MapReduce program runs. It depicts how the MapReduce program runs there by showing how much percentage of map work is done and how much is remaining. Similarly it also shows how much reduce work is remaining and how much is done.

3.5 Performance Evaluation

We will compare the number of bytes written by this map reduce job with increase in size of file. Which in turn will help us to determine whether using Hadoop to overcome big data problem is a feasible solution or not. The performance evaluation is done with keeping in mind the bytes written by map reduce job.

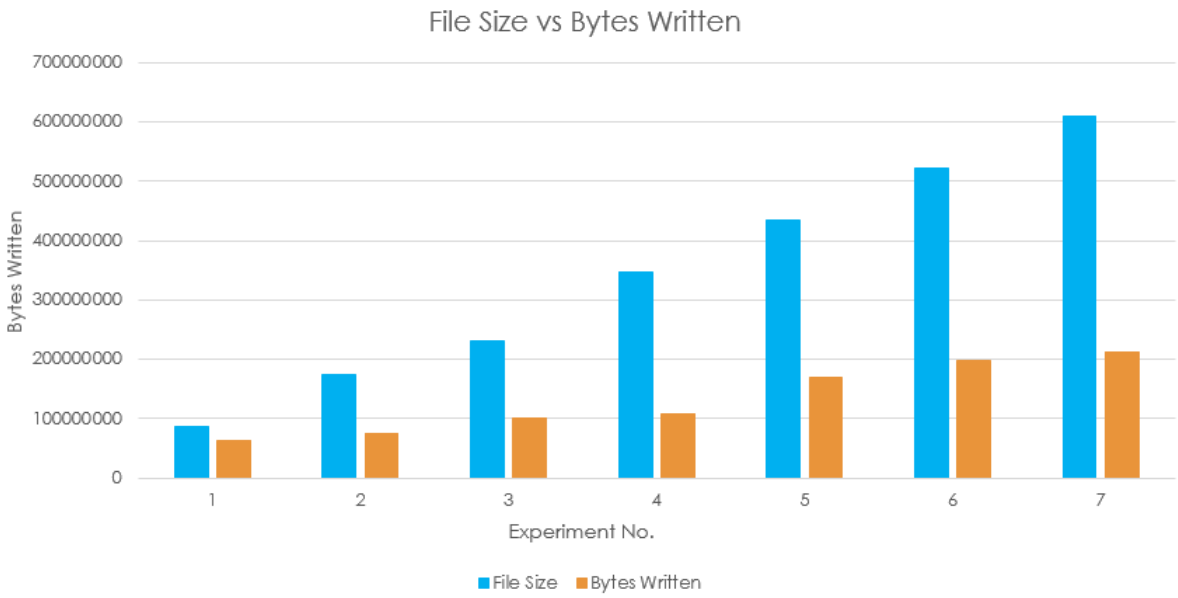

Figure 3.3: Bytes Written

Figure 3.3 shows graph between file size and bytes written. It shows the rate at which the file size is increasing, Bytes Written by MapReduce job is not increasing at that rate. So as a result to overcome the big data processing problem, we can easily see that the bytes written by the MapReduce job is very less as compared to the increase in size of the data.

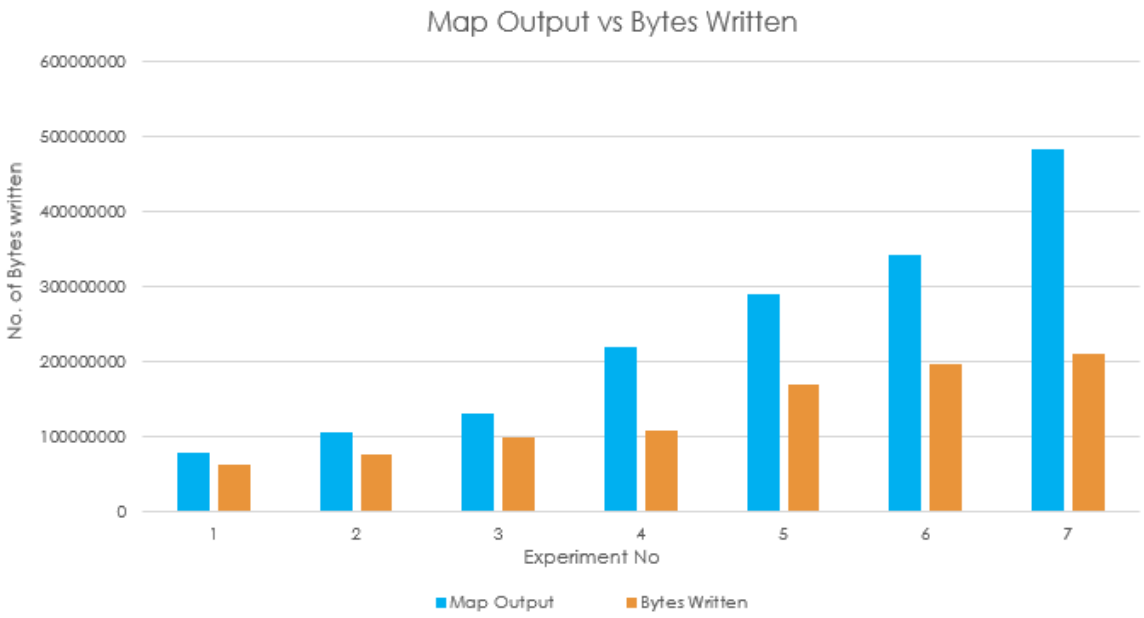

Figure 3.4: Map Output Bytes

Figure 3.4 shows graph between map output and bytes written. Map output actually shows the number of uncompressed bytes given by all map functions. But the bytes written represents how much bytes are written to the file after the reduce step. As in the reduce step, the values are merged and grouped according to key value, so the total bytes produced is significantly lesser than the map output.

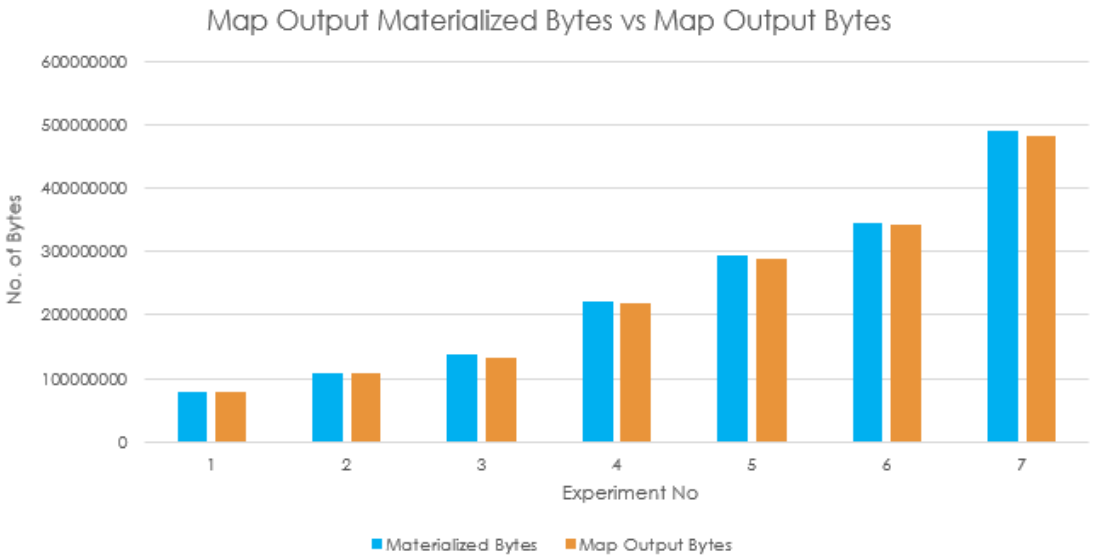

Figure 3.5: Map Output Materialized Bytes vs Map Output Bytes

Figure 3.5 shows graph between map output materialized bytes and map output bytes. The map output shows the actual number of bytes produced by mapper and the materialized map output bytes shows how much bytes actually written to the disk. We can see that the actual number of bytes written to the disk is lesser than the actual bytes produced because the map output shows the uncompressed output but the actually output is compressed and then written to the disk.

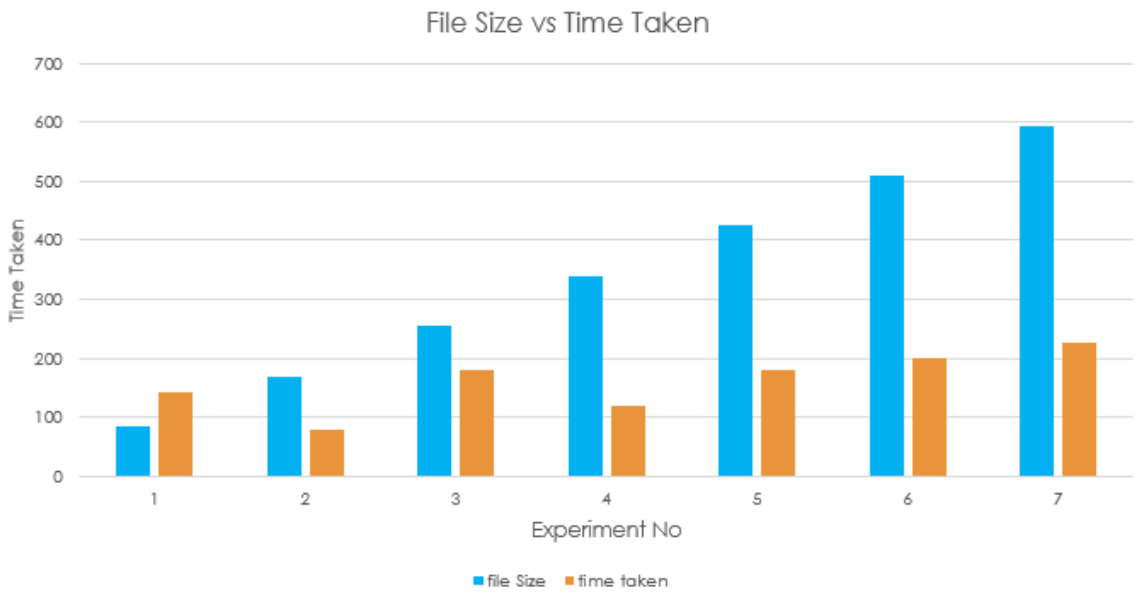

Figure 3.6: File Size vs Time Taken

Figure 3.6 shows graph between file size and time taken. So, it is clear that the rate at which the file size is increasing, the time taken to complete the job is not increasing at that rate. So in big organization which deals with petabytes of data each day need not worry about the time as given the MapReduce architecture provided by Hadoop overcomes this time barrier.

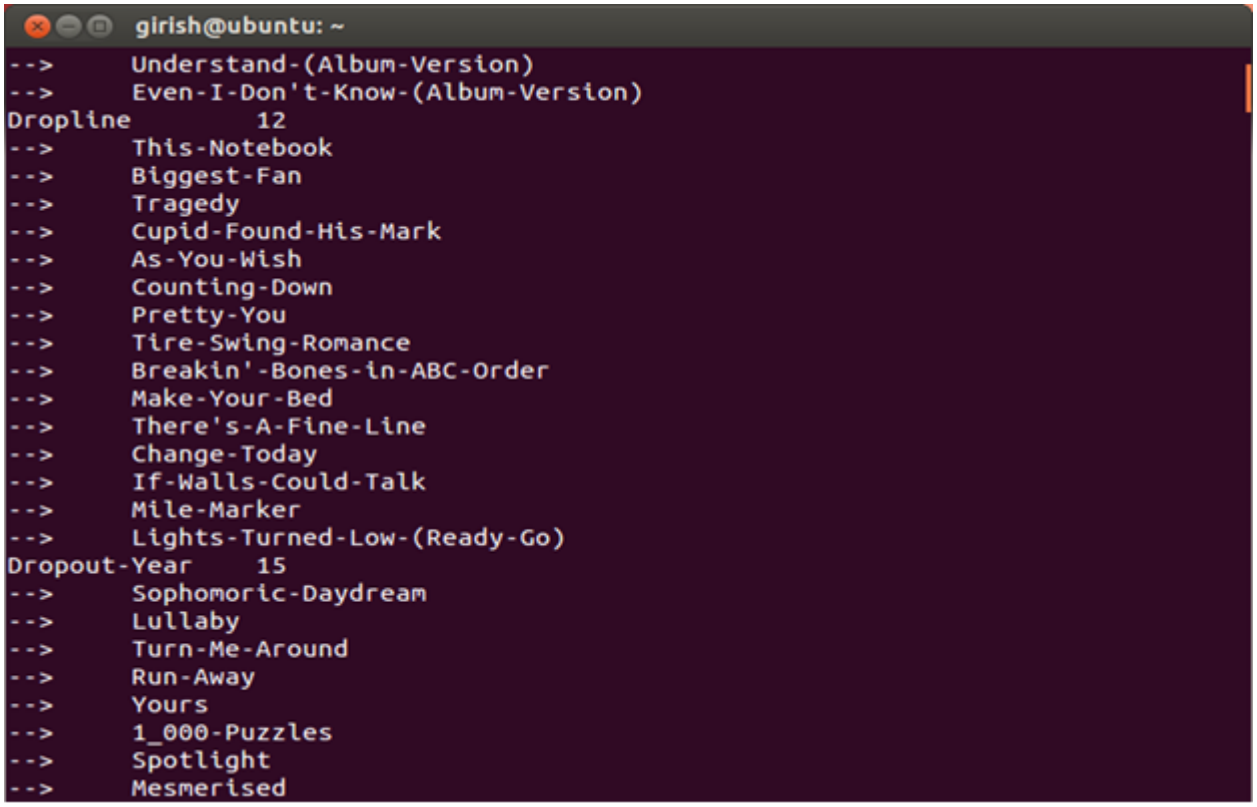

Figure 3.7: Single Node Output

Figure 3.7 shows the desired output stated earlier. As we are concerned with the number of song_titles written by each artist, this figure exactly points out those things. For example the Dropout-Year band which has 15 song_names which are This-Notebook,Biggest-fan, Tragedy, Cupid-Found-His-Mark, As-You-Wish, Counting-Down, Pretty-you, Tire-Swing-Romance, Breaking-Bones-in-ABC-Order, Make-Your-Bed, There's-A-Fine-Line, Change-Today, If-Walls-Could-Talk, Mile-Marker, Lights-Turned-Low-(Ready-Go). So at first all the song_titles done by the artist_name is listed out and then the total count is written.

Chapter 4

Hadoop Security Concerns

4.1 Introduction

information security and privacy of the information still remains as critical issues for Big Data. These issues have not attracted much attention till now. Some researchers pointed out that due to big volumes, variety and variability, Big Data is unattractive for the attackers for now. But Big Data possess many new threats to information security, privacy and protection adopted for traditional security measures, is no longer adequate for Big Data [13].

4.2 Vulnerabilities

A program that is not secure can have access to data which are sensitive such as:- personal profile, age, diseases etc. and corrupt the data which will lead to incorrect results and can be a victim of Denial Of Service.

For increasing performance and scalability of big data different methodologies have been proposed but those methodologies lack in security measures. Those methodologies have been proposed without keeping security aspect in mind. As a result it becomes vulnerable to security threats.

Insecure data storage is one of the biggest vulnerability to big data. The data is stored in distributed datanodes which gives rise to many vulnerabilities such as auto-tiering, facing problem in real time analysis, constant monitoring. As data storage is insecure, so communication between different datanodes becomes insecure and transactional logs of big data also becomes insecure [14].

4.3 Security

Big data has complexities that basically individuals and organizations are ill-equipped to manage. These complexities incorporate security and administration of information all in all. Information governance is the capacity to make data asset that can be trusted by employees, accomplices, and clients, and government associations

Big data originates from numerous data sources that may have distinctive security and administration arrangements. A very much characterized security method must be connected on whatever data administration. Mix of security and administration strategy need joint effort and coordination to share obligations across associations/ gatherings included to verify the responsibility is enforced to the information being utilized.

Many security solution which are widely acceptable can be used to encrypt the data. Nonetheless, various types of data oblige diverse types of security assurances. Applying the same sort of encryption may bring about high cost furthermore, complex procedures.

4.4 Security in Hadoop at present

Hadoop default means consider system as trusted and hadoop client uses local username. In default strategy, there is no encryption in between client

host and Hadoop [15]. In HDFS, all records are put away in clear content i.e plain text and controlled by a focal server called NameNode. Along these lines, HDFS has no security apparatus against storage servers that may peep at information content. Also, Hadoop and HDFS have no solid security model, specifically the communication which are done between DataNodes and DataNodes and communication between clients of DataNodes is not encrypted. To tackle these issues, a few components have been added to Hadoop to look after them. For instance,by appropriate authentication, Hadoop is secured with Kerberos and intensive it, gives shared validation and secures against eavesdropping and replay attacks.

4.5 Cryptography

Cryptography is derived from a greek word with meaning Secret Writing. We use this term to have a secure communication. We use it to send message to other party in secure manner. Cryptography can be broadly divided in to three categories such as follows.

4.5.1 Symmetric-key Cryptography

In Symmetric-key cryptography, one party can send data to another party over an communication channel which is insecure with an assumption that the data cannot be decoded by third party just by simply eavesdropping over the channel. Symmetric-key cryptography uses only a single secret key which is shared by both the concerning parties for encryption and decryption purpose.

4.5.2 Asymmetric-key Cryptography

In symmetric-key cryptography, there is a single key but in asymmetric-key cryptography there are two keys. One is private key and the other is public key.

The data which is encrypted with the public key can be decrypted by with the corresponding private key.

4.5.3 Hashing

This is the third type of encryption system in which fixed length message digest is generated from variable length message. In this scheme the receiver must have the message as well as the digest.

4.6 RSA Cryptosystem

The RSA cryptosystem is named after R. Rivest, A. Shamir, and L. Adleman. RSA cryptosystem is the most widely used public key Cryptosystem.

RSA Algorithm

As RSA is a public key cryptography, it uses two keys that is public key and private key [16]. Public key is used to encrypt the data and the corresponding private key is used to decrypt data. Lets say C is ciphertext and P is the plain text. While encrypting, the following is done

$C=P^e \bmod n$

Where C=cipher text

P=plain text

Both e&n are public

While decrypting, the following is done

$P=C^e \bmod n$

Key Generation :

1. Lets first select two large primes p and q such that p not equal to q.

2. n←p x q

3. ϕ(n)←(p-1) x (q-1)
4. Select e such that $1 < e < \phi(n)$ and e is coprime to ϕ(n).
5. d ← e^{-1} mod ϕ(n)
6. Public key←(e,n)
7. Private Key←d

Encryption

1. Consider one party 'X'wants to send a message to 'Y'securely over an insecure channel.
2. Let us take 'e'be Y's public key. As 'e'is public, X has access to 'e'.
3. To encrypt the message or plain text P, we have to convert the message as an integer which should be in the range $0 < P < n$.
4. Now calculate the cipher text $C=P^e$ mod n

Decryption

1. Let us consider 'X'received 'C',which is the cipher text.
2. Calculate Plain text $P=C^d$ mod n

For example : Let us say we want to send a message using RSA encryption scheme. So at first we have to calculate two large prime numbers. Then calculate the toteint function which will be the multiplication of (p-1) and (q-1). Then generate e and calculate d.

p= 12460968915106246468362709320572320638572441061623070010381828194705350435609497876500578383446504681119055533583746268698561559711301851595629409549069331

q= 10880328353827047039860654961840739738711141418158766748319825592910212178245255071418876384702489135123666618645256570843971951973305558292857791466323459

e=7

d=11621094291701824363900042452575432825770543870691656910539865941082265890959240302083408257435248550049061111874109226936446929602623022152245751320982099953847163518509654787672406942367786932497789764702558950240105935974904799386008860878104633350348331626799016614139111972905531661795488891736626915126 3

n=13557943340318795091216716194671338296732301182473599728963176931262643539452447019097309633674456641723904630519794098092521417869726859177620043207812452280284750998257281407954236340735122482939002703669994645445502353526983651425638462969932220474788011170147408950112918221740955399502392555746166273592 9

Plaintext: I am a student of NIT Rourkela

Ciphertext:37088580485014098458290265018318447853154863164779924208031984281005418413968470695829816673444027151782775119701898806815145285153299527023520144269000360001037527028655103212147576450266175429484623323388958810871474701452063126419707296565955036228464572858712749153171756441549885442385051145889355395878

Public-key cryptosystem such as RSA cryptosystem mainly comprises of the following steps:

1. Selecting two large prime numbers, p and q
2. Picking e
3. Calculating d, which is the inverse of e

Let us first consider how to select or generate p and q such that it would be large enough and unique. We can calculate the value of n which is the multiplication of p and q. Any potential attacker or adversary has access to n because n is public. So

we have to take appropriate and efficient step to prevent the adversary from finding out the value of p and q or making it difficult for the attacker to factorize n by simple methods.

In summary, the steps for selecting those prime number is as follows:

1. Select an odd integer n at random which can be achieved easily by using a random number generator.

2. Pick an integer t < n randomly.

3. Now the next step is to find out whether n satisfies the primality test or not. There are sufficient numbers of probabilistic primality which includes Miller-Rabin, Baillie-PSW primality test etc taking a random number as a parameter. We can have also provable prime numbers which can be generated using Mersenne primes or Fermat primes etc. If n fails the test, then flush the value n and go to step 1.

4. If 'n'has successfully passed the tests, accept n; else, go to step 2.

As the large prime numbers p and q are decided by using probabilistic primality test, so our next focus is to generate key . As stated earlier in the key generation algorithm we have to select an integer 'e'and after that we have to calculate 'd'which is inverse of e modulus n. We have to choose the integer e in such a manner that gcd of (n) and e is 1 that is (n) and e are coprime. Then we have to calculate d which is inverse of 'e'modulus n that is e-1 mod n . There are many algorithms available that will calculate the gcd. If the greatest common divisor is found to be 1 then it is confirmed that they satisfied the constraint of coprimeness. Now we have to calculate the inverse of 'e'modulo n. The algorithm is popularly known as extended Euclid's algorithm.

4.7 Proposed Encryption Method

The proposed method is also an asymmetric-key cryptosystem which uses two keys for encryption and decryption. Public key used for encryption and corresponding private key is used for decryption. [17] proposed a 'n'prime number RSA cryptosystem. [18] proposed another version of RSA. Our proposed scheme has drawn idea from these schemes. In RSA , only two large prime numbers are considered while in our proposed encryption system, we have taken four large prime numbers. As in asymmetric-key cryptography, the point lies in how to make it difficult for the attacker to factorize n which is the multiplication of those four prime numbers. Our proposed system is not only about increasing prime numbers but also we have applied security in terms of public key and private key. In RSA, the public key consists of e and n but we have included another parameter f. And in private key, we have included three other parameters a , b and h.

Key Generation

1. 1. Select four large prime numbers p, q, r and s such that they are all unique and not equal to each other.
2. n←p x q x r x s
3. ϕ(n)←(p-1) x (q-1) x (r-1) x (s-1)
4. Select e such that 1 < e < ϕ(n) and e is coprime to ϕ(n).
5. d ← e^{-1} mod ϕ(n)
6. Select an integer b < ϕ(n)-1
7. Pick another integer a such that b < a < ϕ(n)
8. Pick another integer h such that a < h < ϕ(n)
9. Calculate f=$(b^a)^h$

10. Public key←(e, n, f)
11. Private Key←(d, a, b, h)

Encryption

1. Let's say one party X wants to send a message to Y securely over an insecure channel.
2. Let's take 'e'be Y's public key. 'e'is known to A because 'e'is public key.
3. Now we want to apply our encryption scheme to the message or plain text P. First, we have to convert the plaintext or message to an integer in the range 0 $<$ P $<$ n.
4. Now in the final step calculate cipher text which is C=$(P \times f)^e \bmod n$

Decryption

1. Let us consider C to be the cipher text which is received by Y from X.
2. Now we have to calculate the plain text from the received cipher text which can be derived as:- P= $((b^{\phi(n)-(a \times h)} \bmod n) \times C)^d \bmod n$

So, in this way we can encrypt and decrypt data as per our requirements. The daa can be encrypted using the public key and can be decrypted using the corresponding private key as stated earlier.

Chapter 5

Experiment and Result

Let us say we want to encrypt a message according to our proposed asymmetric-key cryptosystem. Lets say the message is "I am a student of NIT Rourkela".

As per our proposed method, we first have to decide four large prime numbers which are unique and not equal to each other. After finding out those four prime numbers we have to calculate the totient function. The next step is to pick a random natural number e and then calculate the inverse of that number with modulus n where n is the product of those unique prime numbers. Now, we have to pick two other natural numbers which are not larger than the totient function.

5.1 Output of the Proposed Cryptosystem

p=1323668067212056168016060932772486246137164671034381236702121138234539003481103894673261781208753946742506506137903633688647353164894937040872587310905506 1

q=7526308019841990187057091218574258957690413414702202040499081935242474427155580408550276598746413504965701004280748973142943883360183860131480953220087439

r=901910183612082648930422625050318510200839072384467862977364487053251209046889175458189032522954647441479703701010344849530428439372069310431509762358611 3

s=11899769060468942798726362480958554185316613424904168202829680348283668790275946632476987139344331302272896107930233924125068562166687845246155220938647013

ϕ**(n)**=10692097337343956787353953112711451142687905785550815040103030328841012681722067890939173273703455545576585789877041639490026447631102572551195480722632008031126097894441398699199491945780104913333590269616402143266238954041575127491352890739031190820202794671830177354854693787663485211922163564827668732972378302094940529933509067079357723621978931592110008318789436528076651166322264514728266763730231295851644985130746031882170037937520752946054528958047256452036463688943387613846961761530738698052870187366975846301384443584653366206909902323519809060080324089803293012737119538117468469308451498325186150840320

a=215

b=105

h=595

e=11

d=3888035395397802468128710222804164051886511194745750923673829210487640975171661051250608463164892925664213014500742414360009617320400935473161992990048002920409490143433235890617997071192765423030396461678691688460450528742390955451401051177829523934619198062483700856310797740968540077062604932664606811989955746216342010884912388028857354044355975124403639388650704192027873151389914368992097004992811380309689085502089466138970922886371182889474374166562638709831441341433959132307986095102086799291952795406173035018685252212601224075239964481279930567301936032655742913722588922951806716112164181209158600305571

n=10692097337343956787353953112711451142687905785550815040103030328

4101268172206789093917327370345554557658578987704163949002644763110257255119548072263201234352631557615190625995839383388265589519294682433535895034222721672118925955942702457104426741107075234453093493934973445293672196681103657537324132815063213478663640720643896790187137748020938219014339706381979797625036287386169567908565299969031424412082367201730779246372109501856145561646575705004756810760432709864821571066496208095597807194499590537236621782786330144454095266634032861901201850474212066491283600219594638089521184484827150480263298321735 1

Plaintext: I am a student of NIT Rourkela

Ciphertext:8407759042691726806110901519942155176548129672390799350686924633032232277373163000803226193313706414561919091890424833884216987206301747203128586693498287034744563499492676436645486293816954119162377040957109618671466104104260640555681006520735517485869387896480372081777237913780420136358642349823372447842253630356548600228125804083324146794419888993139351476854424703732869527169798609766993726476604160286364939234818537287798108730602074267469397178385832781161884504214905867191054365549233285576037329907133727942799744140519164176699180891446633140073860419612705839136858360722780469389436779787492848606566

5.2 Novel Encryption Scheme for Hadoop

Now we will use our encryption system to the files which are stored in HDFS. As we already discussed that there is no security at all given to data which are stored in Hadoop. So in order to protect data which are sensitive and we dont want an outsider to get access to data, We have to use some security methods. Before storing files in HDFS, the proposed encryption system is applied to each file which encrypts the content of the files and the data can only be retrieved by applying the corresponding private key. Hence, the objective of providing security to the data is achieved.

5.3 Output in Hadoop

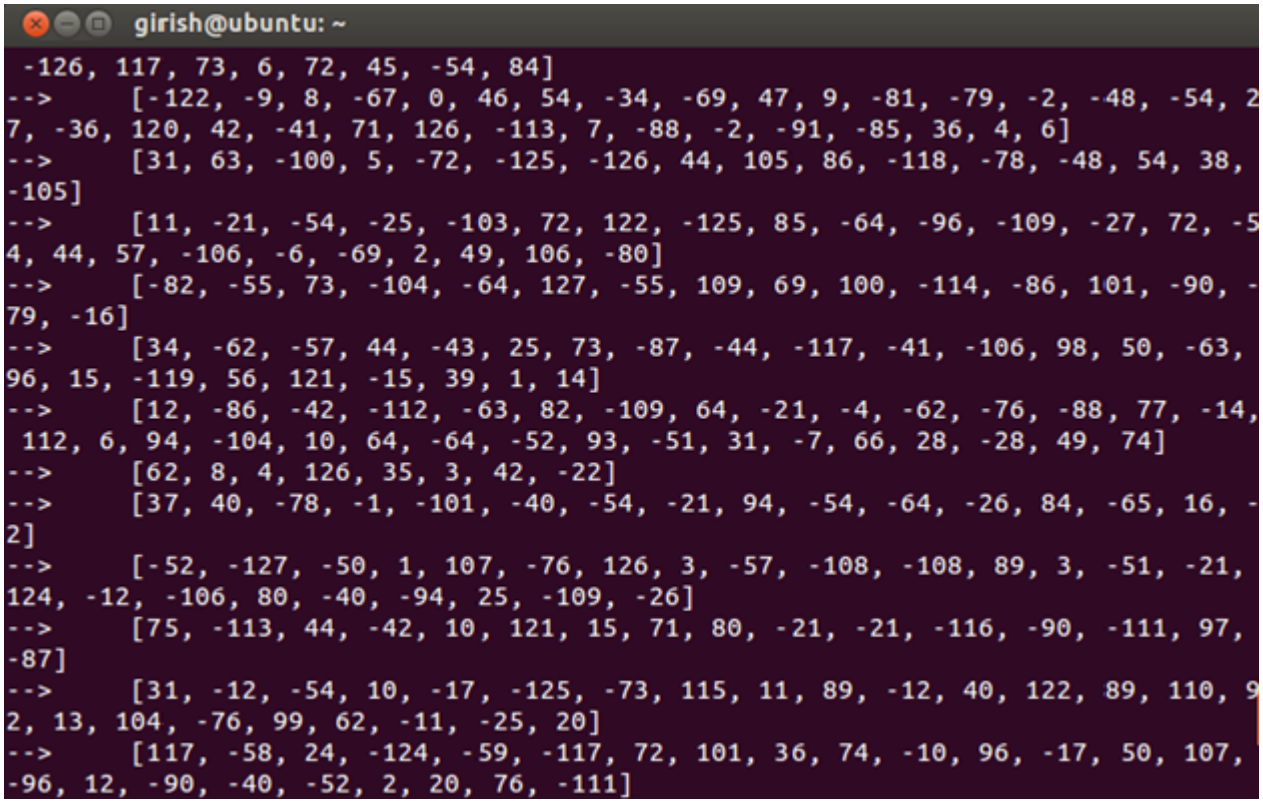

Figure 5.1: Encrypted output of Data

Figure 5.1 shows the content of the file is encrypted and is stored in bytes format rather than in plain text. If an attacker gets access to the cluster and finds the data, it would not create much difference as the data will be in unusable form that is in encrypted form. In order to decrypt it, he must factorize that large numbers into four unique prime numbers which is nearly impossible. In this way we can overcome the security threats associated with hadoop.

5.4 Comparison Between Proposed Scheme and Existing Scheme

The key generation algorithm takes a little more time than RSA algorithm. If we consider encryption of files then our proposed method stands out as it uses four unique prime numbers instead of two as in RSA algorithm. As the number of prime number is increased, it will make the adversary difficult to factorize it. Apart from that there is another natural number each included in both public key and private key. So the overall security increases.

File Size (in Bytes)	Time Taken by Hadoop with out Encryption (in seconds)	Time Taken by Hadoop with Proposed Method (in seconds)	Diffrence (in s)
87040000	143	192	49
174080000	80	188	108
230400000	179	327	148
348160000	119	343	224
435200000	180	385	205
522240000	200	412	212
609280000	226	425	199

Table 5.1: Comparison of proposed method and existing one

Table 5.1 shows how much time Hadoop takes to complete the assigned job with out using any encryption scheme and how much time it takes to complete the map-reduce job after applying the proposed encryption scheme.The file size is shown in no of bytes, the time taken by hadoop to complete the job with out encryption is calculated in seconds, time taken by hadoop to complete the job with our proposed encryption system is calculated in seconds and then the diffrence between thse two

times is taken.

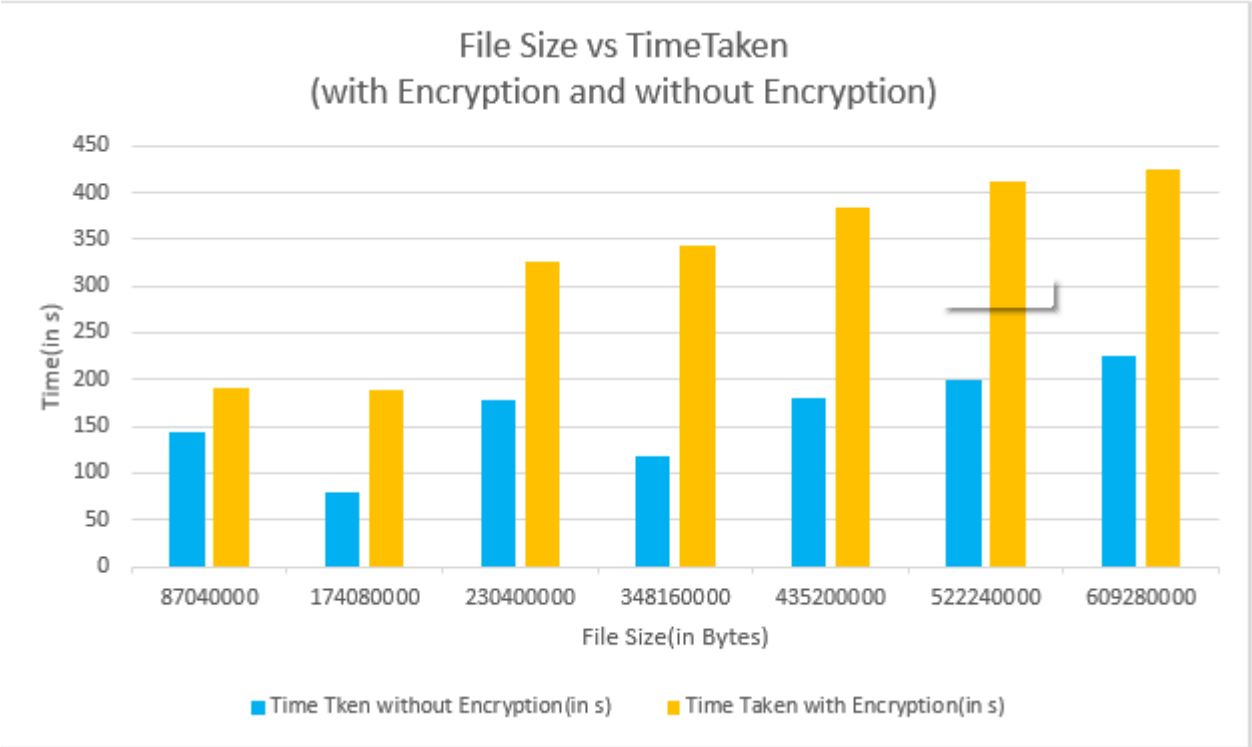

Figure 5.2: File Size vs Time Taken(with and without Encryption)

Figure 5.2 shows the time taken to complete the job increases if we encryption on the files before storing it in HDFS but as compared to the rate at which the file size is increasing, the time taken is not increasing at that rate.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this era of digital world, data is increasing exponentially on daily basis which can't be handled by traditional database management system. So in order to deal with such a huge amount of data given rise to the big data problem. Hadoop is a scalable method to face the challenges caused by big data. As the number of bytes written by the MapReduce job of Hadoop does not increase at the same rate at which data is increasing, so Hadoop is a suitable method to overcome big data problem.

Though Hadoop enables us to overcome challenges faced in industries and institutions due to big data, it has not any security mechanism. The data stored in Hadoop can be compromised by an attacker or eavesdropper. As Hadoop does not provide any security mechanism, the authenticity of data is always at stake. The proposed encryption scheme encrypts the content of files before storing it into HDFS thus by securing it from the attacker. Hence, the data or files now can be stored in Hadoop without worrying about security issues by applying the encryption algorithm on the files before storing it in Hadoop.

6.2 Future Work

In future, we will try to improve the data import mechanism in Hadoop. Moreover, we will try to improve the infrastructure of HDFS for better processing and management of big data.

Bibliography

[1] Apache hadoop en.wikipedia.org/wiki/apache_hadoop.

[2] Sam Madden. From databases to big data. *IEEE Internet Computing*, 16(3):4–6, 2012.

[3] Stephen Kaisler, Frank Armour, J Alberto Espinosa, and William Money. Big data: Issues and challenges moving forward. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 995–1004. IEEE, 2013.

[4] Owen omalley, integrating kerberos into apache hadoop kerberos conference 2010, 26-27 october 2010, mit, usa.

[5] Big data - wikipedia, en.wikipedia.org/wiki/big_data.

[6] Aditya B Patel, Manashvi Birla, and Ushma Nair. Addressing big data problem using hadoop and map reduce. In *Engineering (NUiCONE), 2012 Nirma University International Conference on*, pages 1–5. IEEE, 2012.

[7] Avita Katal, Mohammad Wazid, and RH Goudar. Big data: Issues, challenges, tools and good practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013.

[8] Kamalpreet Singh and Ravinder Kaur. Hadoop: Addressing challenges of big data. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 686–689. IEEE, 2014.

[9] T. white hadoop: The definitive guide. o'reilly media, inc., 2012.

[10] en.wikipedia.org/wiki/mapreduce.

[11] http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn- site/yarn.html.

[12] https://hadoop.apache.org/docs/r2.4.1/hadoop-project- dist/hadoop-hdfs/hdfsuserguide.html.

[13] Rasim Alguliyev and Yadigar Imamverdiyev. Big data: Big promises for information security. In *Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on*, pages 1–4. IEEE, 2014.

[14] Budi Rahardjo Kuspriyanto Marisa Paryasto, Andry Alamsyah. Big-data security management issues. In *International Conference on Information and Communication Technology 2014 (ICoICT 2014), At Bandung*, 2014.

[15] Masoumeh Rezaei Jam, Leili Mohammad Khanli, Morteza Sargolzaei Javan, and Mohammad Kazem Akbari. A survey on security of hadoop. In *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*, pages 716–721. IEEE, 2014.

[16] Cryptography and network security by behrouz a. forouzan.

[17] B Persis Urbana Ivy and Mukesh Kumar PurshotamMandiwa. A modified rsa cryptosystem based on nprime numbers. *International Journal of Engineering And Computer Science ISSN*, pages 2319–7242, 2012.

[18] Sonal Sharma, Jitendra Singh Yadav, and Prashant Sharma. Modified rsa public key cryptosystem using short range natural number algorithm. *International Journal*, 2, 2012.